Amendments to the Claims

Please amend the claims as follows:

1. (Currently Amended): A method for profiling computer program executions in a computer processing system having a processor and a memory hierarchy, comprising the steps of:

executing a computer program;

storing, in a memory array, profile counts for events associated with the execution of the computer program, the memory array being separate and distinct from the memory hierarchy so as to not perturb normal operations of the memory hierarchy;

selecting a plurality of events for profiling;

updating the profile counts for only the selected events; and

assisting compilation of the computer program, based upon the selected profile counts stored in the memory array.

2. (Cancelled).

3. (Previously Presented): The method according to claim 1, wherein said storing and updating steps are performed asynchronously to prevent a decrease of an execution speed of the computer program.

4. (Previously Presented): The method according to claim 1, wherein said updating step is triggered by execution of the events.

5. (Previously Presented): The method according to claim 1, wherein said updating step is triggered by execution of instructions embedded into an instruction stream of the computer program.

6. (Previously Presented): The method according to claim 1, further comprising the step of detecting whether a profile count has exceeded an adjustable predefined threshold.

7. (Presently Presented): The method according to claim 1, further comprising the step of indicating when a profile count has exceeded an adjustable predefined threshold.

8. (Original): The method according to claim 7, wherein said indicating step comprises the step of raising an exception.

9. (Previously Presented): The method according to claim 1, further comprising the steps of:

accumulating profile updates; and

dividing the accumulated profile updates by a threshold fraction.

10. (Previously Presented): The method according to claim 1, further comprising the step of scaling the profile counts to prevent profile information overflow.

11. (Previously Presented): The method according to claim 1, further comprising the step of identifying profile information corresponding to the profile counts using a profiling event identifier.

12. (Original): The method according to claim 11, further comprising the step of addressing the memory array, using the profiling event identifier.

13. (Previously Presented): The method according to claim 1, further comprising the steps of:

generating the profile counts using profile counters associated with the events; and

maintaining the profile counters in a set-associate manner.

14. (Original): The method according to claim 13, further comprising the step of selecting a profile counter to be evicted from the memory array based upon a predefined replacement, when a number of profiling events assigned to an associative class of events is exceeded.

15. (Original): The method according to claim 14, wherein the replacement strategy is based upon one of least-recently-used and first-in-first-out.

16. (Previously Presented): The method according to claim 1, further comprising the step of supporting read operations from the memory array in an off-line optimization of the program.

17. (Currently Amended) The method according to claim 1, further comprising the step of assisting optimization of the program, based upon the profile counts stored in the ~~profile matrix.~~memory array.

18. (Original): The method according to claim 17, wherein said assisting step is performed during at least one of dynamic binary translation and dynamic optimization of the computer program.

19. (Original): The method according to claim 18, wherein the dynamic binary translation and dynamic optimization of the computer program results in translated and optimized code, respectively, the translated and optimized code comprising instructions groups which pass control therebetween.

20. (Original): The method according to claim 19, further comprising the step of identifying frequently executed paths of the computer program, by instrumenting exits from the instruction groups with a profiling instruction that indicates a unique group exit identifier.

21. (Original): The method according to claim 19, further comprising the step of extending the instruction groups along a frequently executed path.

22. (Original): The method according to claim 1, wherein the memory hierarchy includes data and instruction caches, and the memory array is separate and distinct from

the memory hierarchy so as to not perturb normal operations of the data and instruction caches.

23. (Currently Amended): An apparatus for profiling computer program executions in a computer processing system having a processor and a memory hierarchy, the apparatus comprising:

a memory array adapted to store profile counts for events associated with execution of the computer program, said memory array being separate and distinct from the memory hierarchy so as to not perturb normal operations of the memory hierarchy; ~~and~~

a controller adapted to select the events for profiling and to update the profile counts of the selected events stored in said memory array; and

a scaling circuit adapted to scale the profile counts to prevent profile information overflow;

wherein the computer processing system assists compilation of the computer program, based upon the profile counts stored in the memory array.

24. (Original): The apparatus according to claim 23, wherein said memory array and said controller are adapted to asynchronously store and update the profile counts, respectively, to prevent a decrease of an execution speed of the computer program.

25. (Original): The apparatus according to claim 23, wherein said controller is adapted to update the profile counts as the events are executed.

26. (Original): The apparatus according to claim 23, wherein said controller is adapted to update the profile counts based upon instructions embedded into an instruction stream of the computer program.

27. (Original): The apparatus according to claim 23, further comprising a comparator circuit adapted to detect whether a profile count has exceeded an adjustable predefined threshold.

28. (Original): The apparatus according to claim 23, further comprising an indicating circuit for indicating when a profile count has exceeded an adjustable predefined threshold.

29. (Original): The apparatus according to claim 28, wherein said indicating circuit is adapted to raise an exception when the profile count has exceeded the adjustable predefined threshold.

30. (Original): The apparatus according to claim 23, further comprising:

an accumulation circuit adapted to accumulate the updated profile counts; and

a dividing circuit adapted to divide an accumulated value of the updated accumulated profile counts by a threshold fraction.

31. (Cancelled).

32. (Previously Presented): The apparatus according to claim 23, wherein profile information corresponding to the profile counts is identified using a profiling event identifier.

33. (Original): The apparatus according to claim 32, wherein the memory array is addressed using the profiling event identifier.

34. (Original): The apparatus according to claim 23, further comprising profile counters for generating the profile counts, said profile counters being associated with an event in a set-associate manner.

35. (Previously Presented): The apparatus according to claim 34, further comprising a replacement circuit adapted to select a profile counter to be evicted from the memory array based on a predefined replacement strategy, when a number of profiling events assigned to an associative class is exceeded.

36. (Previously Presented): The apparatus according to claim 35, wherein the predefined replacement strategy is based upon one of least-recently-used and first-in-first-out.

37. (Original): The apparatus according to claim 23, wherein the memory hierarchy includes data and instruction caches, and said memory array is separate and distinct from the memory hierarchy so as to not perturb normal operations of the data and instruction caches.

38. (Original): The method according to claim 1, wherein said method is implemented by a program storage device readable by machine, tangibly embodying a program of instructions executable by the machine to perform said method steps.

39. (Previously Presented): The apparatus according to claim 23, further comprising wherein the computer processing system assists optimization of the computer program, based upon the profile counts stored in the memory array.

40. (Currently Amended): A method for profiling computer program executions in a computer processing system having a processor and a memory hierarchy, comprising the steps of:

executing a computer program; ~~and~~

storing, in a single memory array, a plurality of event-specific profile counts for a plurality of events associated with the execution of the computer program, the memory array being separate and distinct from the memory hierarchy so as to not perturb normal operations of the memory hierarchy;

selecting a plurality of events for profiling;

updating the profile counts for only the selected events; and

storing, in a global counter, a global profile count comprising a total of the plurality of event-specific profile counts;

wherein each of the plurality of event-specific profile counts is uniquely assigned to count only one of the plurality of events; and

wherein profile information associated with the profile counts describes a typical execution path of the computer program.

41. (Previously Presented): The method according to claim 40, further comprising the step of optimizing the computer program during at least one of static and dynamic compilation using the profile information.

42. (Previously Presented): The method according to claim 40, wherein the memory array is arranged as a two-way set associative array.